

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 1 – מעודכן לתאריך 24.11.2025  
עמוד 1 מתוך 8



מתרגל ממונה על התרגיל: איל רביד, eyal.ravid@campus.technion.ac.il

תאריך ושעת הגשה: 21/12/2025 בשעה 23:59

אופן ההגשה: בזוגות. אין להגיש ביחידים. (אלא באישור מתרגל אחראי של הקורס).

## הנחיות כלליות:

- שאלות על התרגיל יש לפרסם באתר הפיאצה של הקורס תחת לשונית "wet\_1":
  - האתר: <https://piazza.com/technion.ac.il/winter2026/234218>, נא לקרוא את השאלות של סטודנטים אחרים לפני שמפרסמים שאלה חדשה, למקרה שנשאלה כבר.
- נא לקרוא את המסמך "נהלי הקורס" באתר הקורס. בנוסף, נא לקרוא בעיון את כל ההנחיות בסוף מסמך זה.
- בפורום הפיאצה ינוהל FAQ ובמידת הצורך יועלו תיקונים כהודעות נעוצות (Pinned Notes). תיקונים אלו מחייבים.
- התרגיל מורכב משני חלקים: יבש ורטוב.
  - לאחר קריאת כלל הדרישות, מומלץ לתכנן תחילה את מבני הנתונים על נייר. דבר זה יכול לחסוך לכם זמן רב.
  - לפני שאתם ניגשים לקודד את פתרונכם, ודאו כי יש לכם פתרון העומד בכל דרישות הסיבוכיות בתרגיל. תרגיל שאינו עומד בדרישות הסיבוכיות יחשב כפסול.
  - את הפתרון שלכם מומלץ לחלק למחלקות שונות שאפשר לממש (ולבדוק!) בהדרגתיות.
  - המלצות לפתרון התרגיל נמצאות באתר הקורס תחת: "Programming Tips Session".
  - המלצות לתכנות במסמך זה אין מחייבות, אך מומלץ להיעזר בהן.
  - חומר התרגיל הינו כל החומר שנלמד בהרצאות ובתרגולים עד אך לא כולל עצי דרגות.
- העתקת תרגילי בית רטובים תיבדק באמצעות תוכנת בדיקות אוטומטית, המזהה דמיון בין כל העבודות הקיימות במערכת, גם כאלו משנים קודמות. לא ניתן לערער על החלטת התוכנה. התוכנה אינה מבדילה בין מקור להעתק! אנא הימנעו מהסתכלות בקוד שאינו שלכם.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת:  
goldshtein@campus.technion.ac.il



### הקדמה:

בעקבות באג במערכת סאפ, נמחקו כל הנתונים על הסטודנטים והקורסים בפקולטה. בזכות הידע הנרחב שצברתם בקורס, מזכירות הפקולטה מבקשת מכם, הסטודנטים לבנות מערכת חדשה בשם TechSystem לניהול מאגר הסטודנטים והקורסים. למזכירות הפקולטה דרישות מאוד גבוהות והמערכת שתבנו תצטרך לעמוד בדרישות סיבוכיות נוקשות.



### סימונים לצורכי סיבוכיות:

נסמן ב- $n$  את מספר הסטודנט במערכת.

ב- $m$  את מספר הקורסים במערכת.

ב- $n_{courseId}$  את מספר הסטודנטים הרשומים לקורס בעל המזהה  $courseId$  אם אין קורס כזה אפשר להתייחס לערך זה כ-1.

### דרוש מבנה נתונים למימוש הפעולות הבאות:

`techSystem()`

מאתחלת מבנה נתונים ריק. תחילה אין במערכת סטודנטים או קורסים.

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן:  $O(1)$  במקרה הגרוע.

---

`virtual ~techSystem()`

הפעולה משחררת את המבנה (כל הזיכרון אותו הקצאתם חייב להיות משוחרר).

פרמטרים: אין

ערך החזרה: אין

סיבוכיות זמן:  $O(n \cdot m)$  במקרה הגרוע.

---

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 1 – מעודכן לתאריך 24.11.2025

עמוד 3 מתוך 8



`StatusType addStudent(int studentId)`

הפעולה מוסיפה למבנה נתונים סטודנט בעל `studentId` מזהה עם 0 נק"ז. הסטודנט לא רשום לאף קורס.

פרמטרים:

`studentId` מזהה מזהה הסטודנט שצריך להוסיף.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $studentId \leq 0$ .
FAILURE	אם קיים כבר סטודנט עם מזהה <code>studentId</code> .
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן:  $O(\log n)$  במקרה הגרוע.

`StatusType removeStudent(int studentId)`

הסטודנט בעל המזהה `studentId` סיים את התואר ולכן צריך להוציאו מהמערכת. אם הסטודנט רשום לאחד או יותר קורסים אז לא נוציא אותו מהמערכת והוא עוד לא יסיים את התואר.

פרמטרים:

`studentId` מזהה הסטודנט.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $studentId \leq 0$ .
FAILURE	אם אין סטודנט עם מזהה <code>studentId</code> או שהסטודנט רשום לקורסים.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן:  $O(\log n)$  במקרה הגרוע.

`StatusType addCourse(int courseId, int points)`

קורס בעל מזהה ייחודי `courseId` מתווסף למערכת, סיום הקורס מזכה את הסטודנט המסיים ב `points` נק"ז. תחילה לקורס לא רשומים סטודנטים.

פרמטרים:

`courseId` מזהה הקורס שצריך להוסיף.  
`points` מספר הנק"ז שהקורס מזכה סטודנט שעבר אותו.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $courseId \leq 0$ או $points \leq 0$ .
FAILURE	אם קיים כבר קורס במזהה <code>courseId</code> במערכת.
SUCCESS	במקרה של הצלחה.

סיבוכיות זמן:  $O(\log m)$  במקרה הגרוע.

`StatusType removeCourse(int courseId)`

עקב חוסר ביקוש, מבטלים את הקורס בעל המזהה `courseId` ומוציאים אותו מהמערכת. אם יש סטודנטים הרשומים לקורס, אז הקורס לא יוצא מהמערכת ולא יבוטל.

פרמטרים:

`courseId` מזהה הקורס שמבקשים להסיר.

ערך החזרה:

ALLOCATION_ERROR	במקרה של בעיה בהקצאה/שחרור זיכרון.
INVALID_INPUT	אם $courseId \leq 0$ .

# מבני נתונים 234218 חורף תשפ"ו



גיליון רטוב מספר 1 – מעודכן לתאריך 24.11.2025

עמוד 4 מתוך 8

אם אין קורס עם מזהה $courseId$ או שיש סטודנטים הרשומים לקורס בעל המזהה $courseId$	FAILURE
במקרה של הצלחה.	SUCCESS
סיבוכיות זמן: $O(\log m)$ במקרה הגרוע.	

**StatusType enrollStudent(int studentId, int courseId)**

סטודנט בעל המזהה studentId נרשם לקורס בעל המזהה courseId.

פרמטרים:

מזהה הסטודנט שרוצה להירשם.	studentId
מזהה הקורס אליו הסטודנט נרשם.	courseId

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $studentId \leq 0$ או $courseId \leq 0$	INVALID_INPUT
אם אין סטודנט עם מזהה studentId או אין קורס עם מזהה courseId במערכת או שהסטודנט כבר רשום לקורס אליו הוא מנסה להרשם.	FAILURE
במקרה של הצלחה.	SUCCESS
סיבוכיות זמן: $O(\log n + \log m)$ במקרה הגרוע.	

**StatusType completeCourse(int studentId, int courseId)**

סטודנט בעל המזהה studentId השלים את חובותיו בקורס בעל המזהה courseId ולכן הוא מקבל את הנק"ז של הקורס ומוצא מהקורס.

פרמטרים:

מזהה הסטודנט שהשלים את הקורס.	studentId
מזהה הקורס שאותו הסטודנט השלים.	courseId

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $studentId \leq 0$ או $courseId \leq 0$	INVALID_INPUT
אם אין קורס במזהה courseId או אין סטודנט בעל המזהה studentId או שהסטודנט בעל המזהה studentId לא רשום לקורס עם המזהה courseId.	FAILURE
במקרה של הצלחה.	SUCCESS
סיבוכיות זמן: $O(\log m + \log n_{courseId})$ במקרה הגרוע.	

**StatusType awardAcademicPoints(int points)**

דיקן הפקולטה החליט לתת בונוס לכל הסטודנטים בפקולטה של points נק"ז במתנה. הבונוס תקף אך ורק לסטודנטים שנמצאים במערכת בזמן הקריאה לפונקציה ולא משפיע על הנק"ז של סטודנטים שיצטרפו בעתיד.

פרמטרים:

מספר הנק"ז שהדיקן מעניק לסטודנטים.	points
------------------------------------	--------

ערך החזרה:

במקרה של בעיה בהקצאה/שחרור זיכרון.	ALLOCATION_ERROR
אם $points \leq 0$ .	INVALID_INPUT
במקרה של הצלחה.	SUCCESS

סיבוכיות זמן:  $O(1)$  במקרה הגרוע.

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 1 – מעודכן לתאריך 24.11.2025  
עמוד 5 מתוך 8



---

`output_t < int > getStudentPoints(int studentId)`

מחזיר את מספר הנק"ז שצבר הסטודנט בעל מזהה `studentId`.

פרמטרים:

`studentId` מזהה הסטודנט שצריך להחזיר את מספר הנק"ז שצבר.

ערך החזרה:

ALLOCATION\_ERROR במקרה של בעיה בהקצאה/שחרור זיכרון.

INVALID\_INPUT אם `studentId ≤ 0`.

FAILURE אם אין סטודנט בעל מזהה `studentId`.

SUCCESS במקרה של הצלחה, במקרה זה יוחזר גם מספר הנק"ז של הסטודנט.

סיבוכיות זמן:  $O(\log n)$  במקרה הגרוע.

**דוגמה הרצה:**

```
addCourse(1, 3): SUCCESS
addCourse(2, 5): SUCCESS
addStudent(1): SUCCESS
addStudent(5): SUCCESS
getStudentPoints(1): SUCCESS, Value: 0
enrollStudent(1, 1): SUCCESS
removeStudent(1): FAILURE
removeCourse(1): FAILURE
getStudentPoints(1): SUCCESS, Value: 0
completeCourse(1, 1): SUCCESS
completeCourse(1, 2): FAILURE
removeCourse(1): SUCCESS
getStudentPoints(1): SUCCESS, Value: 3
awardAcademicPoints(10): SUCCESS
getStudentPoints(1): SUCCESS, Value: 13
removeStudent(1): SUCCESS
addStudent(100): SUCCESS
getStudentPoints(100): SUCCESS, Value: 0
```

**סיבוכיות מקום:**

סיבוכיות המקום הדרושה עבור מבנה הנתונים היא  $O(n + m + \sum_{i \text{ is a course}} n_i)$  במקרה הגרוע. כלומר בכל רגע בזמן הריצה, צריכת המקום של מבנה הנתונים תהיה לינארית בסכום מספרי הסטודנטים, הקורסים במערכת ומספר הסטודנטים בכל הקורסים יחד כולל כפילויות.

סיבוכיות המקום הנדרשת עבור כל פעולה (כלומר, זיכרון "העזר" שכל פעולה משתמשת בו) אינה מצוינת לכל פעולה לחוד, אך אסור לעבור את סיבוכיות המקום הדרושה שמוגדרת לכל המבנה.

**ערכי החזרה של הפונקציות:**

כל אחת מהפונקציות מחזירה ערך מטיפוס `StatusType` שייקבע לפי הכלל הבא:

- תחילה, יוחזר `INVALID_INPUT` אם הקלט אינו תקין.
  - אם לא הוחזר `INVALID_INPUT`:
  - בכל שלב בפונקציה, אם קרתה שגיאת הקצאה/שחרור יש להחזיר `ALLOCATION_ERROR`. מצב זה אינו צפוי אלא באחד משני מקרים (לרוב): באמת השתמשותם בקלט גדול מאוד ולכן המבנה ניצל את כל הזיכרון במערכת, או שיש זליגת זיכרון בקוד.
  - אם קרתה שגיאה אחרת, כפי שמצוין בכל פונקציה, יש להחזיר מיד `FAILURE` מבלי לשנות את מבנה הנתונים.
  - אחרת, יוחזר `SUCCESS`.
- חלק מהפונקציות צריכות להחזיר בנוסף עוד פרמטר (`int` או `bool`), לכן הן מחזירות אובייקט מטיפוס `output_t<T>`. אובייקט זה מכיל שני שדות: הסטטוס (`__status`) ושדה נוסף (`__ans`) מסוג `T`.

במקרה של הצלחה (SUCCESS), השדה הנוסף יכיל את ערך החזרה, והסטטוס יכיל את SUCCESS. בכל מקרה אחר, הסטטוס יכיל את סוג השגיאה והשדה הנוסף לא מעניין.

שני הטיפוסים (output\_t<T>, StatusType) ממומשים כבר בקובץ "wet1util.h" שניתן לכם כחלק מהתרגיל.

קיים קונסטרקטור של output\_t<T> מ-T ו-m StatusType כך שניתן פשוט לכתוב בפונקציות הרלוונטיות:

return 7;

return StatusType::FAILURE;

### הנחיות ודגשים כלליים:

#### חלק יבש:

- החלק היבש מהווה חלק מהציון על התרגיל כפי שמצוין בנהלי הקורס.
- לפני מימוש הפעולות בקוד יש לתכנן היטב את מבני הנתונים והאלגוריתמים ולוודא כי באפשרותכם לממש את הפעולות בדרישות הזמן והזיכרון שלעיל.
- החלק היבש חייב להיות מוקלד.
- הגשת החלק הרטוב מהווה תנאי הכרחי לקבלת ציון על החלק היבש, כלומר, הגשה בה יתקבל אך ורק חלק יבש תגרור ציון 0 על התרגיל כולו.
- יש להכין מסמך הכולל תיאור של מבני הנתונים והאלגוריתמים בהם השתמשתם בצירוף הוכחת סיבוכיות הזמן והמקום שלהם. חלק זה עומד בפני עצמו וצריך להיות מובן לקורא גם לפני העיון בקוד. אין צורך לתאר את הקוד ברמת המשתנים, הפונקציות והמחלקות, אלא ברמה העקרונית. חלק יבש זה לא תיעוד קוד.
- ראשית הציגו את מבני הנתונים בהם השתמשתם. רצוי ומומלץ להיעזר בצירוף.
- לאחר מכן הסבירו כיצד מימשתם כל אחת מהפעולות הנדרשות. הוכיחו את דרישות סיבוכיות הזמן של כל פעולה תוך כדי התייחסות לשינויים שהפעולות גורמות במבני הנתונים.
- הוכיחו שמבנה הנתונים וכל הפעולות עומדים בדרישת סיבוכיות המקום.
- החסמים הנתונים בתרגיל הם לא בהכרח הדוקים ולכן יכול להיות שקיים פתרון בסיבוכיות טובה יותר. מספיק להוכיח את החסמים הדרושים בתרגיל.
- רמת פירוט: יש להסביר את כל הפרטים שאינם טריוויאליים ושחשובים לצורך מימוש הפעולות ועמידה בדרישות הסיבוכיות. אין לדון בפרטים טריוויאליים (הפעילו את שיקול דעתכם בקשר לזה, ושאלו את האחראי על התרגיל אם אינכם בטוחים). אין לצטט קטעים מהקוד כתחליף להסבר. אין צורך לפרט אלגוריתמים שנלמדו בכתה. כמו כן, אין צורך להוכיח תוצאות ידועות שנלמדו בכתה, אלא מספיק לציין בבירור לאיזו תוצאה אתם מתכוונים. אין (וגם אין צורך) להשתמש בתוצאות של עצי דרגות והלאה.
- **על חלק זה לא לחרוג מ-8 עמודים.**
- והכי חשוב **keep it simple!**

### חלק רטוב:

- מומלץ לממש תחילה את מבני הנתונים בצורה הכללית ביותר ורק אז לממש את הפונקציות הנדרשות בתרגיל.
- אנו ממליצים בחום על מימוש **Object Oriented**, ב-C++, מימוש כזה יאפשר לכם להגיע לפתרון פשוט וקצר יותר לפונקציות אותן עליכם לממש ויאפשר לכם להכליל בקלות את מבני הנתונים שלכם (זכרו שיש תרגיל רטוב נוסף בהמשך הסמסטר).
- פקודת הקימפול שמורצת בgradescope הינה: g++ -std=c++14 -DNDEBUG -Wall -o main.out \*.cpp
- חתימות הפונקציות שעליכם לממש ומספר הגדרות נמצאים בקובץ TechSystem26a1.h.
- אין לשנות את הקבצים main26a1.cpp ו-wet1util.h אשר סופקו כחלק מהתרגיל, ואין להגיש אותם. ישנה בדיקה אוטומטית שאין בקוד שימוש ב-STL, ובדיקה זו נופלת אם מגישים גם את main26a1.cpp.
- את שאר הקבצים ניתן לשנות, ותוכלו להוסיף קבצים נוספים כרצונכם, ולהגיש אותם.
- העיקר הוא שהקוד שאתם מגישים יתקמפל עם הפקודה לעיל, כאשר מוסיפים לו את שני הקבצים wet1util.h ו-main25b1.cpp.
- עליכם לממש בעצמכם את כל מבני הנתונים (למשל אין להשתמש במבנים של STL ואין להוריד מבני נתונים מהאינטרנט). **כחלק מתהליך הבדיקה אנו נבצע בדיקה ידנית של הקוד ונוודא שאכן מימשתם את מבני הנתונים שבהם השתמשתם.**
- בפרט, אסור להשתמש ב-std::vector, std::pair, או כל אלגוריתם של STL, רשימה מלאה של הספריות להן אסור לעשות include נמצאת בקובץ dont\_include.txt.
- ניתן להשתמש במצביעים חכמים (Smart pointers כמו shared\_ptr), בספריית math או בספריית exception.

# מבני נתונים 234218 חורף תשפ"ו

גיליון רטוב מספר 1 – מעודכן לתאריך 24.11.2025

עמוד 8 מתוך 8



- חשוב לוודא שאתם מקצים/משחררים זיכרון בצורה נכונה (מומלץ לוודא עם valgrind). לא חייבים לעבוד עם מצביעים חכמים, אך אם אתם מחליטים כן לעשות זאת, לוודא שאתם משתמשים בהם נכון. (תזכרו שהם לא פתרון קסם, למשל, כאשר יוצרים מעגל בהצבעות). שימו לב שהעתקת מצביעים חכמים היא איטית מאוד ולכן יכולה לגרום ל-timeout. עדיף להעביר shared\_ptr כשצריכים by reference.
- שגיאות של ALLOCATION\_ERROR בד"כ מעידות על זליגה בזיכרון.
- על הקוד להתקמפל ולעבור את כל הבדיקות שמפורסמות לכם ב-gradescope. הטסטים שמורצים באתר מייצגים את הבדיקות אותן נריץ בנתינת הציון, כאשר פרסמנו 5 מתוך 50.
- אותם טסטים שבgradescope גם מפורסמים כקבצי קלט ופלט, יחד עם סקריפט בשם run\_tests.py שנכתב בשביל python 3.6 ומעלה, המאפשר לבדוק את הקוד שלכם. מומלץ לבדוק את התרגיל לוקאלית לפני שמגישים.
- במידה ויש timeout על אחד מהטסטים זה יחשב ככשלון בטסט. עליכם לכתוב קוד יעיל במידת הסביר. אין לדאוג מכך יותר מידי, הרוב המוחלט של הפתרונות שעומד בסיבוכיות עומד גם בזמני הריצה.
- שימו לב:** התוכנית שלכם תיבדק על קלטים רבים ושונים מקבצי הדוגמא הנ"ל. יחד עם זאת הטסטים האלו מייצגים מבחינת אורך ואופן היצירה שלהם את השאר.

## אופן ההגשה:

הגשת התרגיל מתבצעת דרך אתר ה- gradescope של הקורס.

## חלק הרטוב:

יש להגיש רק את קבצי הקוד שלכם (לרוב קבצי .h, .cpp, בלבד אפשר גם להגיש אותם כקובץ zip) ב- gradescope תחת המשימה "wet 1 code part".

## חלק היבש:

יש להגיש קובץ PDF אשר מכיל את הפתרון היבש ב- gradescope תחת המשימה "wet 1 dry".

## **שימו לב שהחלק היבש חייב להיות מוקלד.**

- שימו לב כי אתם מגישים את כל שני החלקים הנ"ל, במטלות השונות.**
- לאחר שהגשתם, יש באפשרותכם לשנות את התוכנית ולהגיש שוב. ההגשה האחרונה היא הנחשבת.
- הערכת הציון שמופיעה ב-gradescope אינה ציונכם הסופי על המטלה. הציון הסופי יתפרסם רק לאחר ההגשות המאוחרות של משרתי המילואים.
- במידה ואתם חושבים שישנה תקלה מהותית במערכת הבדיקה ב-gradescope נא להעלות זאת בפורום הפיאצה ונטפל בה בהקדם.

## דחיות ואיחורים בהגשה:

- דחיות בתרגיל הבית תינתנה אך ורק לפי תקנון הקורס.
- 5 נקודות יורדו על כל יום איחור בהגשה ללא אישור מראש. באפשרותכם להגיש תרגיל באיחור של עד 5 ימים ללא אישור. תרגיל שיוגש באיחור של יותר מ-5 ימים ללא אישור מראש יקבל 0.
- במקרה של איחור בהגשת התרגיל יש עדיין להגיש את התרגיל אלקטרונית דרך אתר הקורס.
- בקשות להגשה מאוחרת יש להפנות למתרגל האחראי בלבד בכתובת goldshtein@campus.technion.ac.il. לאחר קבלת אישור במייל על הבקשה, מספר הימים שאושרו לכם נשמר אצלנו. לכן, אין צורך לצרף להגשת התרגיל אישורים נוספים או את שער ההגשה באיחור.

**בהצלחה!**